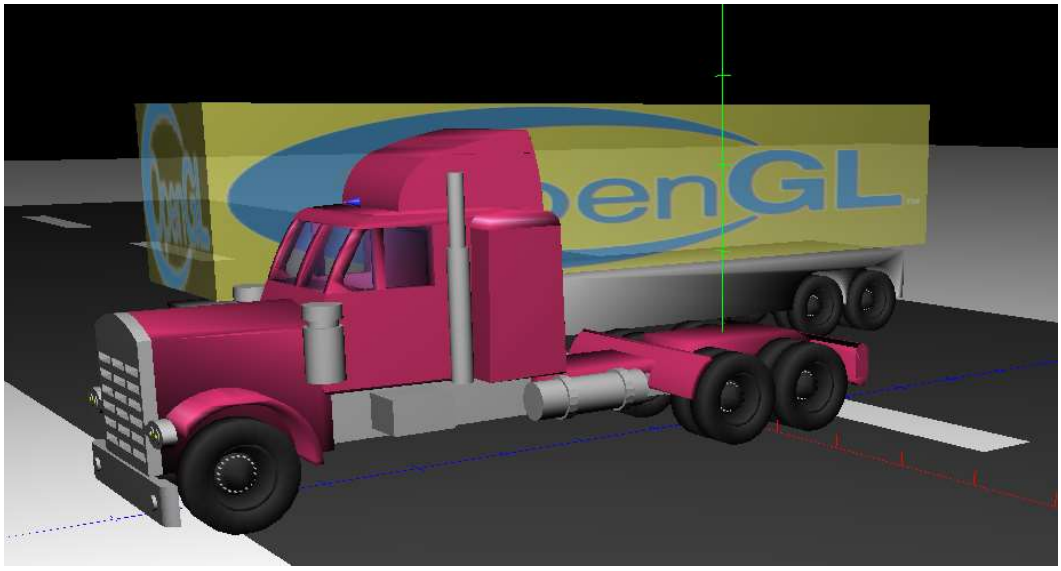
PrakCG Übungsaufgabe

- Texturen und Blending –



Die Szene enthält transparente Objekte. Das sind die Fensterscheiben des Trucks sowie die Container, welche sich auf den Trailern befinden. Die Objekte sind als `TRUCK_GLASS` sowie `TRAILER_CARGO` benannt.

Damit Objekte überhaupt transparent erscheinen, müssen folgende Voraussetzungen erfüllt sein:

- Beim Zeichnen dieser Objekte müssen die gerasterten Fragmente einen Alphawert kleiner 1 besitzen, d.h. die Materialdefinition bzw. die Farbe und/oder die Texturen müssen entsprechend spezifiziert sein. Im Template erfolgt das Setzen des Materials für die Objekte mittels *`cg_object3D::setMaterial(...)`*.
- Das Blending muss in OpenGL aktiviert und die Blendfunktion muss geeignet spezifiziert sein:

  ```
  glEnable(GL_BLEND);
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  ```
- Sichtbare transparente Objekte müssen vor den Hintergrund gezeichnet werden. Das bedeutet, dass die transparenten Flächen erst dann gezeichnet werden, wenn der Framebuffer bereits alle dahinterliegenden Objekte enthält. Die Reihenfolge der transparenten Flächen muss dabei in Sichtrichtung der Kamera von hinten nach vorn erfolgen. Das erfordert also eine Sortierung der transparenten Objekte nach der Entfernung von der Kamera.

Schauen Sie sich zuerst das fertige Programm `complete.exe` an. Bewegen Sie die Kamera, so dass Sie die Trailer durch die Fenster des Trucks sehen und dann den Truck durch die Trailer. Die hinter den transparenten Objekten im Vordergrund befindlichen Fahrzeuge sind immer zu sehen.

Bedienung

- Tasten l,L - Beleuchtung an/aus
- Tasten t,T - Texturierung an/aus
- Tasten b,B - Blending an/aus.

Betrachten Sie nun einen Container aus verschiedenen Richtungen. Auch hier sind die „hinteren“ Flächen immer korrekt zu sehen. Damit das funktioniert, muss das Zeichnen der Container-Flächen auch von hinten nach vorn erfolgen.

Aufgabe 1:

Machen Sie die Fenster des Trucks transparent. Achten Sie auf die korrekte Darstellung, d.h. alle hinter den Scheiben befindlichen Fahrzeuge und die soliden Teile des Trucks sollen aus jeder Perspektive durch die Scheiben hindurch sichtbar sein, siehe Titelbild.

- Weisen Sie dem Material des Objekts `TRUCK_GLASS` ein Alpha kleiner als 1.0 zu, also z.B. 0.5.
- Aktivieren Sie das OpenGL-Blending vor dem Zeichnen der Fenster in `CTruck::draw()`.
- Deaktivieren Sie das Blending nach dem Zeichnen der Fenster.

Hinweis: Innerhalb von `CTruck::draw()` werden die Scheiben zuletzt gezeichnet. Damit wird die o.g. Reihenfolge eingehalten.

Aufgabe 2:

Machen Sie nun in der gleichen Art und Weise den Container transparent (Objekt `TRUCK_CARGO`)! Kontrollieren Sie das Ergebnis, indem Sie die Szene aus unterschiedlichen Perspektiven betrachten. Die anderen Fahrzeuge müssen zu sehen sein, wenn sie sich hinter einem transparenten Container befinden. Das ist nicht immer der Fall.

Passen Sie in `drawScene()` die Zeichenabfolge für die drei Fahrzeuge in Abhängigkeit von der aktuellen Kameraposition an. Verwenden Sie folgende Positionen zum Berechnen der Entfernungen:

- Kameraposition: `cg_globState::cameraPos[]`
- Position Truck: `CTruck::getPos()`
- Position Trailer*i*: `CTrailer*i*::getPos()`.

Zeichnen Sie dann die Fahrzeuge von “hinten” nach “vorn”. Hierfür gibt es unterschiedliche Herangehensweisen. Im Template wird folgende gewählt:

- Das Feld `indexes[]` enthält anfangs die Objekt-**Indizes** in der Reihenfolge **0**-truck, **1**-trailer1, **2**-trailer2.
- Das Feld `distances[]` enthält die zugehörigen **Entfernungen** zur Kamera.
- Das Feld `indexes[]` wird nach den absteigenden Entfernungen mit dem „bubblesort“-Algorithmus sortiert. Es werden dabei nicht die Elemente in `distances[]` vertauscht, sondern in `indexes[]`.
- Danach erfolgt das Zeichnen der Objekte anhand ihrer **Reihenfolge** in `indexes[]`.

Aufgabe 3:

Texturieren Sie das Objekt `TRUCK_CARGO` in `CTrailer::draw()`!

Hinweis: Das Objekt `TRUCK_CARGO` enthält bereits Texturkoordinaten, die pro Vertex definiert sind. Sie mappen die Textur jeweils geeignet auf jede Fläche des Quaders.

Es sind bereits zwei Texturen geladen: „container.bmp“ und „opengl.bmp“. Die erste Textur enthält nur RGB-Farben, die zweite zusätzlich einen Alphakanal. Sie ist überall dort transparent, wo keine Schrift vorhanden ist.

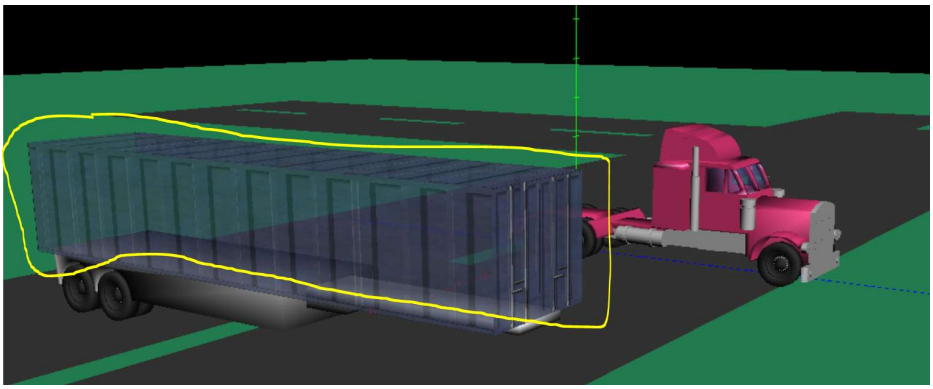
Eine der beiden Texturen kann verwendet werden, indem vor dem Zeichnen des CARGO-Objektes die Texturierung aktiviert und die entsprechende Textur gebunden wird, siehe Übung 11. Die Auswahl der Textur erfolgt beim Anlegen der Trailer vor `drawScene()`.

Experimentieren Sie dazu mit den unterschiedlichen Texturmodi (`GL_REPLACE`, `GL_MODULATE`, `GL_DECAL` und `GL_BLEND`), siehe auch https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glTexEnv.xml. Stellen Sie dazu vor dem Binden der Textur den gewünschten Modus ein.

Vergessen Sie nicht, die Texturierung nach dem Zeichnen des Objektes wieder zu deaktivieren.

Aufgabe 4:

Untersuchen Sie Ihr Ergebnis auf Darstellungsfehler im Bereich des Containers. Diese entstehen offensichtlich durch die konstante Abfolge der 6 Flächen beim Zeichnen des Quaders. Das folgende Bild zeigt solche Fehler: Die hinteren Flächen des Containers sind teilweise nicht zu sehen.



Offensichtlich muss man die Tiefensortierung entsprechend Aufgabe 2 nicht pro Objekt, sondern pro Fläche vornehmen. Wenn sich Flächen gegenseitig durchdringen, dann müssten diese sogar an der Schnittkante geteilt werden. Dies würde jedoch einen größeren Eingriff ins Template bedeuten.

Glücklicherweise gibt es für konvexe Objekte wie unseren Container eine einfachere Möglichkeit. Sie basiert auf dem BackFace Culling. Damit lassen sich alle weiter entfernten Flächen zuerst zeichnen. Das sind genau die Flächen, von denen wir die Rückseiten sehen.

Überlegen Sie sich die Vorgehensweise zur korrekten Darstellung des Containers, siehe `complete.exe` und implementieren Sie diese in `CTrailer::draw()`.

Zur Erinnerung:

- BackFace Culling aktivieren / deaktivieren:
  `glEnable(GL_CULL_FACE); / glDisable(GL_CULL_FACE);`
- zu entfernende Seite festlegen:
  `glCullFace(GL_FRONT); / glCullFace(GL_BACK);`